BeyondTrust

AD Bridge
Integration Guide

TC:10/16/2020

Table of Contents

Introduction to the AD Bridge Integration Guide

AD Bridge Enterprise joins Unix and Linux computers to Active Directory so that you can centrally manage all your computers from one source, authenticate users with the highly secure Kerberos 5 protocol, control access to resources, and apply group policies to non-Windows computers. The result: vastly simplified account management, improved security, reduced infrastructure costs, streamlined operations, and easier regulatory compliance.

AD Bridge Enterprise provides graphical tools to manage Linux and Unix information in Active Directory. However, it can be useful to access and modify the information programmatically. For this purpose, AD Bridge Enterprise provides scripting objects that can be used by any programming language that supports the Microsoft Common Object Model, or COM. The scripting objects provide dual interfaces that can be used by languages that use COM early binding, such as C++ and C#, and by languages that use Idispatch, such as VBScript and Jscript.

This document describes the AD Bridge Enterprise scripting objects and explains how to use them.

Standard Integrations with AD Bridge

AD Bridge integrates using only standards at the OS level. For example, nsswitch and PAM. All standard OS functions like **whoami**, **id**, and **passwd** operate with AD Bridge users as if they were a standard user. Integration with a number of different password vaulting solutions at various levels have been tested in the past with AD Bridge, and in addition, at the time of writing this we are not aware of any issues with using bridged accounts with 3rd party password vaulting solutions that use the operating system function calls in a standards based way.

NFS Integration for Systems Administrators

This section assumes you are a systems administrator who knows how to manage shared files and folders on Linux, Unix, and Windows computers, including configuring the Linux and Unix file servers to run NFS and to comply with your IT security policy.

Instructions on how to set up NFS are beyond the scope of this document.

Requirements

The following prerequisites must be in place:

- Root access to the Linux or Unix file server where you want to run Samba and AD Bridge Enterprise.
- AD Bridge Enterprise 8.5.5 or later.
- DNS capable of resolving FQDN of the NFS server and clients
- The Linux or Unix computer must be connected to Active Directory with AD Bridge.

For instructions on how to join a domain, please see the AD Bridge Installation Guide at www.beyondtrust.com/docs/ad-bridge/getting-started/installation.

Server Setup

1. Install AD Bridge 8.5.5 (or later).
2. Add NFS Service Principal Name(SPN) to the machine. This step should be done before we join the domain to make sure the right SPNs are added to the machine account and the keytab file. If the system is already joined you need to run the domainjoin again after the new **ServicePrincipalName** is set: **/opt/pbis/bin/config ServicePrincipalName "host" "nfs"**.
3. Join the domain: **domainjoin-cli join pbisdemo.com Administrator**.
4. Check keytab file content for SPNs: **/opt/pbis/bin/klist -e -k /etc/krb5.keytab**. Look for:

```
4 nfs/rhel7@PBISDEMO.COM
4 nfs/RHEL7@PBISDEMO.COM
4 nfs/rhel7.pbisdemo.com@PBISDEMO.COM
4 nfs/RHEL7.PBISDEMO.COM@PBISDEMO.COM
```

5. Install NFS Server: **yum install nfs-utils nfs4-acl-tools**.
6. Start NFS Server: **systemctl start nfs-server**.
7. Export Shares: **vim /etc/exports**. Making sure the folders exist, add entries like:

```
/export/data/test *(rw,sec=sys:krb5:krb5i:krb5p,sync,nohide)
```

```
/export/data/department *(rw,sec=sys:krb5:krb5i:krb5p,sync,nohide)
```

8. Export filesystem: **exportfs -ra**.

Client Setup - CentOS 6

1. Install AD Bridge Enterprise 8.5.5 (or later).
2. Before domain join: **/opt/pbis/bin/config ServicePrincipalName "host" "cifs" "nfs"**.
3. Join the domain: **domainjoin-cli join pbisdemo.com Administrator**.
4. Install nfs4 acl: **yum install nfs4-acl-tools**.
5. Enable NFS4 by setting **SECURE_NFS** to *yes* in **/etc/sysconfig/nfs**: **SECURE_NFS="yes"**.
6. Services restart: **service rpcidmapd restart**.
7. Configure autofs to mount it:

```
vim /etc/auto.test
* -fstype=nfs4,rw,sec=krb5,intr,hard,exec,insecure,no_subtree_check,wsize=4096,rsize=4096
rhel7.pbisdemo.com:/export/data/&
```

8. Restart autofs: **service autofs restart**.

Now each user should have a krb5 ticket to access the shares when they authenticate. If you su to a user as root you need to run **kinit** to generate that users own krb5 ticket.

Client Setup - CentOS 7

Same steps as CentOS 6 until step 4:

1. Install nfs4 acl: **yum install nfs4-acl-tools nfs-utils**.
2. Configure autofs:

```
vim /etc/auto.test
* -fstype=nfs4,rw,sec=krb5,intr,hard,exec,insecure,no_subtree_check,wsize=4096,rsize=4096
rhel7.pbisdemo.com:/export/data/&
```

***Note:** Always use FQDNs when mounting NFS shares for SPNs to match the keytab entries.*

***Note:** Autofs is not needed; it is provided as a use case. Manual mount example:*

```
mount -t nfs4 -o sec=krb5 rhel7.pbisdemo.com:/export/data/& /mnt -vvv
```

Samba Integration for Systems Administrators

This section assumes you are a systems administrator who knows how to manage shared files and folders on Linux, Unix, and Windows computers, including configuring the Linux and Unix file servers to run Samba and to comply with your IT security policy.

Instructions on how to set up Samba are beyond the scope of this document.

> For information about installing and configuring Samba, please see the Samba documentation at https://www.samba.org/samba/docs/.

Requirements

The following prerequisites must be in place:

- Root access to the Linux or Unix file server where you want to run Samba and AD Bridge Enterprise.
- AD Bridge Enterprise 6.0.8330 or later
- The Linux or Unix computer must be connected to Active Directory with AD Bridge.

> For instructions on how to join a domain, please refer to the *AD Bridge Enterprise Installation Guide*.

- Samba 3.6 or later
- Samba 4.X support requires AD Bridge 8.5.2 or higher
- Samba package must support ADS security. AD Bridge relies on ADS security in a Samba and AD Bridge configuration.

> For more information, please see Setting up Samba as a Domain Member at https://wiki.samba.org/index.php/Setup_Samba_as_an_AD_Domain_Member.

Install Files

AD Bridge includes a tool to install the files necessary to use Samba: **/opt/pbis/bin/samba-interop-install –install**.

Run the tool with the install option:

- AD Bridge **idmapper** plug-in for Winbind replaces Samba's version.
- **libwbclient** is replaced with the AD Bridge version.
- The machine password is synchronized in Active Directory.
- The AD Bridge version of **libwbclient** communicates directly with the AD Bridge authentication service, instead of Winbind.

Upgrade

If any of the following occur then it is recommended to reinstall the **samba-interop** tool. This is to correct any synchronization that might have been broken.

- Samba is updated
- operating system is updated
- AD Bridge is upgraded
- joining a new domain
- rejoining your existing domain

Configure Samba on a Linux or Unix Computer

1. On the Linux or Unix computer that is running Samba, add the following settings to the global section of the Samba configuration file (Typically located at **/etc/samba/smb.conf**):

```
[global]
security = ADS
workgroup = DEMO
realm = DEMO.COM
machine password timeout = 0
```

The ADS value for the security setting is required. Replace the values of **workgroup** and **realm** with the values for the network. The workgroup is the computer's NetBIOS domain name. The realm is the computer's Active Directory domain.

***Note:** If the machine password option is not added to the **smb.conf** and set to **0**, Samba will change the machine account password without notifying the AD Bridge authentication service, leaving AD Bridge unable to connect to the domain.*

2. If an alternate hostname is used, then set that hostname as the NetBIOS name: **netbios name = CENTOS-TEST**.
3. Create a new section to define a shared resource and constrain access to the Active Directory group **pbis_group**. Limit write access to **pbisadmin**:

```
[testshare]
comment = This is a test share
path = /share
valid users = +DEMO\pbis_group
write list = DEMO\pbisadmin
```

4. Run the **testparm** command to make sure **smb.conf** contains no syntax errors.
5. Make sure the path exists and permissions for the share are set:

```
mkdir /share
chmod 750 /share
chown DEMO\\pbisadmin:DEMO\\pbis_group /share
```

6. Restart Samba: **systemctl restart smbd**.
7. Create a dns entry for the Samba server: **/opt/pbis/bin/update-dns**.

The computer is now ready to access the share from a Windows computer and log on with an Active Directory account.

Home Shares

Example of using homes with AD Bridge home directories:

```
[homes]
   comment = Home Directory of User %U in Domain %D
   path = /home/%D/%U
   browseable = no
   create mask = 640
   directory mask = 0750
   valid users = %U
```

Debug

To help troubleshoot, turn on Samba logging by adding the following settings to the global section of the Samba configuration file, **smb.conf**:

```
[global]
...
#Debugging settings:
log level = 10
debug pid = true
log file = /var/log/samba/smbd.log
max log size = 50 # max 50KB per log file, then rotate
```

Troubleshoot the Samba Integration

1. Check firewall ports are open for Samba. Make sure that at least the following ports are open for use by Samba:
    - 137/udp: used by nmbd
    - 138/udp: used by nmbd
    - 139/tcp: used by smbd
    - 445/tcp: used by smbd
2. Check the folder permissions of the share.
3. Verify that the machine password in **secrets.tdb** is up to date by running: **net ads testjoin**. A successful result will look like this: *Join is OK*.
4. Test user authentication locally with **smbclient**: **smbclient -L 127.0.0.1 -U DEMO\\pbisadmin**.

Net ADS Testjoin Failed

If there is an issue, manually compare the machine password that is stored in **secrets.tdb** (location varies across the Linux distributions) with the machine password that is used by AD Bridge.

Use **tdbtool** to check the machine password in **secrets.tdb**:

```
# cd /var/lib/samba/private/; ls
msg.sock  passdb.tdb  secrets.ldb  secrets.tdb
# tdbtool
tdb> open secrets.tdb
tdb> dump
```

To list AD Bridge password run: **/opt/pbis/bin/lsa ad-get-machine password**.

The passwords must match. If they do not, resolve the mismatch by re-running the AD Bridge Samba **interop tool**. The tool resynchronizes the machine password in **secrets.tdb** with the machine password AD Bridge set in Active Directory. Samba will need to be restarted for the change to take effect. Make sure **machine password timeout = 0** is set to prevent this from occurring.

Authentication Failure - NT_STATUS_LOGON_FAILURE

If **smbclient** returns *NT_STATUS_LOGON_FAILURE* as in the below results:

```
[root@cen73 ~]# smbclient -L 127.0.0.1 -U pbisadmin
Enter DEMO\pbisadmin's password:
session setup failed: NT_STATUS_LOGON_FAILURE
```

Make sure that the SAM account name exactly matches the first component of the UPN used Samba, as shown in the following examples.

1. Check the SAM account name by running:

   ```
   [root@cen73 ~]# /opt/pbis/bin/lsa ad-get-machine account | grep SAM
   SAM Account Name: CEN1234-SU1AY3B$
   ```

2. Compare the SAM account name with the first component of the UPN used by Samba in the logs:

   ```
   [root@cen73 ~]# tail -f log.smbd | grep kerberos_kinit_password
   kerberos_kinit_password CEN123456789123456789$@DEMO.COM failed: Client not found in Kerberos
   database
   ```

If the SAM account name and the first component of the UPN do not match, resolve the mismatch by doing the following:

1. Make sure the host name is 15 characters or less.
2. Make sure there are no computer accounts in AD that have the same SAM account name but a different DNS suffix.
3. Leave the domain with **--deleteAccount**.
4. Rejoin the domain.
5. Try **smbclient** test again.

Fix Error Code 40022: Failed to Refresh Machine TGT

If you get an error in the log that looks something like the following entries (the time stamps and the machine name have been removed), you must add the machine password timeout option to the global section of **smb.conf** and set it to **0** to integrate AD Bridge with Samba:

```
lsassd[1722]: 0x7fafc3ff7700:Error:
Failed to refresh machine TGT [Error code: 40022]
lsassd[1722]: 0x7fafc3ff7700:Error:
Failed to refresh machine TGT [Error code: 40022]
```

***Note:** If the machine password option is not added to the **smb.conf** and set to **0**, Samba will change the machine account password without notifying the AD Bridge authentication service, leaving AD Bridge unable to connect to the domain.*

Configure Aliases in Samba Tips

Use a Username Map for Aliases

With Samba 3.0.25, you can use the non-SAM account aliases of AD Bridge Enterprise by including a user name map:

- Add **username map = /etc/samba/users.map** to the global section of **smb.conf**.
- Create an **/etc/samba/users.map** file.
- In the **users.map** file, add an entry for each aliased user in the following form: **!alias = DOMAIN\user**.

To make an alias for an Active Directory group, use the form **!alias = @DOMAIN\group**. The exclamation point triggers Samba to stop processing on the first matching alias, preventing issues with multiple alias matches from wildcards.

For more information about how to add users to a user name map, please see the Samba documentation .

Enable SSO with Active Directory in Java Application Servers

This chapter explains how to set up single sign-on (SSO) with Active Directory in Java web applications.

Understand Integrated Windows Authentication

Integrated Windows Authentication was introduced with the Microsoft Windows 2000 operating system. It is based on the SPNEGO, Kerberos, and NTLMSSP protocols. The SPNEGO protocol is used between the web browser and the web server to negotiate the type of authentication that will be performed, usually either Kerberos or NTLMSSP. Kerberos is the preferred authentication mechanism. Both Kerberos and NTLMSSP are secure protocols that allow computers to authenticate a user over a non-secure channel. For web sites, this means that the Secure Socket Layer (SSL) protocol does not need to be enabled during the authentication phase.

Why Use Integrated Windows Authentication?

Integrated Windows Authentication improves the overall security of a network because the user must log on by using his or her username and password only once. All subsequent accesses by that user to resources, such as web sites, file systems, and network printers are automatically authenticated with cached security tokens. Using Integrated Windows Authentication has the benefit of a centralized user account database stored in Active Directory. This is more secure and more efficient than duplicating user names and passwords in configuration files across server computers.

Kerberos, NTLMSSP versus Basic Authentication

Integrated Windows Authentication uses the SPNEGO, Kerberos and NTLM authentication protocols. Not all browsers are capable of understanding these protocols. Another authentication protocol, Basic Authentication, is understood by all web browsers; it works by simply transferring the username and password across the network from the web browser to the web server. The drawback of using Basic Authentication is that without SSL encryption, anyone can intercept the network communication and easily find out a user's login name and password. Therefore, Basic Authentication should be used only for sites that are protected with SSL encryption.

Authentication versus Authorization

The term authentication refers to the process of proving a user's identity. Authorization, on the other hand, takes place after authentication and is used to limit the users or groups to perform only the actions that they are allowed or authorized to perform. Integrated Windows Authentication provides only the authentication mechanism while Windows authorization is usually accomplished using Access Control Lists (ACL).

Requirements to Use AD Bridge with Active Directory in SSO

- Root access to the Linux computer
- The Linux computer is joined to Active Directory with Likewise AD Bridge 6.1 later
- Administrative access to the Active Directory for creating a service account if one does not exist
- The Linux computer is running Apache Tomcat Server version 5.5 or later
- The server is running JRE 1.5.0 or higher
- The AD Bridge application integration package is installed

IMPORTANT!

Configuring Java application servers is a complex procedure. Before you deploy your configuration to a production web server, implement and test it in a test environment. Before you change your application server's configuration, read and understand the documentation that accompanies the application server. Before you change a file, make a backup copy.

Components

The following Likewise components relevant to Apache integration are installed at **/opt/pbis/{lib, lib64}**.

Component Name	Description
lwjplatform.jar	AD Bridge Platform Library
lwservlets.jar	AD Bridgeauthentication modules, including Servlet Filter and JAAS Module.
wtomcat.jar	AD Bridge authentication modules specific to implementing the Tomcat authentication valve. This component is not needed if you use the authentication filter described in this document.
jna.jar	Java Native Access Library patched for UCS-2 support.
commons-logging- 1.1.1.jar, **log4j-1.2.16.jar**	Logging facilities
commons-codec- 1.4.jar	Base64 and other encoding routines
commons-net-2.2.jar	Network utilities

Install the Authentication Components

The components from the integration package must be installed. Typically, an Apache Tomcat installation uses the following environment variables:

Environment Variable	Value
CATALINA_HOME	Example: **/usr/share/tomcat8**
CATALINA_BASE	Example: **/var/lib/tomcat8**

To integrate your web application with the Likewise servlet filter you must install the following components in **${CATALINA_HOME}/webapps/<web application>/web-inf/lib**:

- lwservlets.jar
- lwjplatform.jar
- jna.jar
- commons-logging-1.1.1.jar
- log4j-1.2.16.jar
- commons-codec-1.4.jar
- commons-net-2.2.jar

Symbolic links can be created to these jar files from the target directory.

Generate Kerberos Keytab File

The Kerberos keytab file is necessary to authenticate incoming requests. It contains an encrypted, local copy of the host's key. It is important to protect the file with proper file access permissions. The file must be readable by the user or group under which the Apache Tomcat server is running, typically tomcat on most Linux systems. No other user should be able to read or modify the file.

To generate the keytab file:

- Find the server name of the web site that will require authentication. You can use the server name from Active Directory.
- You need to know the fully qualified name of the domain to which the Linux system is joined.
- You need to decide where to save the generated keytab file.

The steps below use a sample Apache user account name named **tomcat**, a sample server name of **myserver**, a sample full domain name of **MYDOMAIN.COM**, and a sample Kerberos keytab file named **/etc/krb5_myserver.keytab**. You must substitute the correct names from your system and configuration.

1. Select a user in Active Directory for the application server. If you create a new user, make sure to set the password for the account to **never expire**. Also, make sure **Use DES Encryption types for this account** is not checked in the user account properties in Active Directory. In this example, we are using the user: **MYDOMAIN\tomcat**.
2. Generate keytab entry on your Windows domain controller for the default HTTP service principal:

```
# ktpass /out c:\krb5_myserver.keytab /pType KRB5_NT_PRINCIPAL /crypto RC4- HMAC-NT /princ
HTTP/myserver.mydomain.com@MYDOMAIN.COM /mapuser  tomcat@MYDOMAIN.COM /mapop set /pass password
```

***Note:** If using Windows Server 2003, and prompting for password via **/pass ***, the domain controller may store an incorrect password preventing login.*

3. Copy the file to the target Linux machine and change the group ownership of the keytab file: **# chown tomcat:tomcat /etc/krb5_myserver.keytab**.
4. Set appropriate file permissions of the keytab file:

```
# chgrp tomcat /etc/krb5.keytab
# chmod g+r /etc/krb5.keytab
```

***Note:** If you choose not to create a separate keytab but rather merge the generated data with default keytab file (typically **/etc/krb5.keytab**), you must provide read access to the file to the local tomcat user.*

5. Set the **KRB5_KTNAME** environment variable of the Tomcat process. This can be set in the beginning of the Tomcat initialization script at **/etc/init.d/tomcat**: **export KRB5_KTNAME=FILE:/etc/krb5_myserver.keytab**.

Change the Application Web Descriptor

Now you need to add filter and filter-mapping sections to the web descriptor of your web application (**web-inf/web.xml**). It must be done for every web application you are planning to protect with the Likewise servlet filter. The example below demonstrates how your

configuration may look (remember to replace **MYDOMAIN** with your short domain name):

```
<filter>
  <filter-name>LikewiseAuth</filter-name>
  <filter-class>com.likewise.auth.filter.spnego.LikewiseNegotiateFilter</filter-class>
  <init-param>
     <param-name>deny-role</param-name>
     <param-value>MYDOMAIN\guests</param-value>
  </init-param>
  <init-param>
     <param-name>allow-role</param-name>
     <param-value>MYDOMAIN\domain^users</param-value>
  </init-param>
  <init-param>
     <param-name>remote-address-accept-filter</param-name>
     <param-value>10.100.0.0/24</param-value>
  </init-param>
  <init-param>
     <param-name>remote-address-accept-filter</param-name>
     <param-value>10.100.1.0,255.255.255.0</param-value>
  </init-param>
</filter>

<filter-mapping>
   <filter-name>LikewiseAuth</filter-name>
   <url-pattern>/*</url-pattern>
</filter-mapping>
```

The above configuration ensures that only members of the **MYDOMAIN\domain^users** group can access the web pages from this application. Users who belong to the **MYDOMAIN\guests** group will be denied access. It is possible to configure multiple deny and allow roles. The user is checked for membership in the deny roles before being checked in the allow roles.

Note: *Role names are case-sensitive. Always use upper-case register for the domain name component and lower-case for the group names. Use* ***^*** *in place of white spaces in group names.*

The **remote-address-accept-filter** configuration parameter can be used to specify IP addresses in the CIDR format. For example, **10.100.0.0/24**. It can also be in the form of IP Address with Subnet mask. For example, **10.100.0.0,255.255.255.0**. If at least one **remote-address-accept-filter** parameter is specified, the servlet performs authentication only of the requests whose remote IP Address is in the range of the permitted addresses. The filter first tries to obtain the IP address of the client from X-Forwarded-For HTTP header. The X-Forwarded-For (XFF) HTTP header field is a de facto standard for identifying the originating IP address of a client connecting to a web server through an HTTP proxy or load balancer. If the header is not set the filter gets the client IP address directly from the TCP/IP socket. If the client IP address is not in the configured range of accepted IP addresses the filter passes the HTTP request to the web application unauthenticated. As a result, the user principal object will not be set in the HTTP session. The access control logic of the web application must decide how to treat unauthenticated HTTP requests. It may choose to reject access, or redirect the request to a different URL, or allow restricted access to the application resources.

Edit the Java policy file

The JVM running the application server puts certain restrictions on the type of operations the web application code is allowed to perform. The authentication filter must be able to read configuration files, and load and execute native libraries in the **/opt/likewise** directory. For this reason, the Java policy file needs to be extended. This step can be skipped if the Java security manager is disabled

in your application server configuration. Below is a sample policy file for Tomcat6. This file can be saved as **/var/lib/tomcat6/conf/policy.d/20sample.policy**.

Consult your application server documentation to find the location of the policy file. Note that name **myapp** below must be substituted with the real name of your web application. You will also need to adjust the **${catalina.base}** variable appropriately since it is only available in Tomcat:

```
grant codeBase "file:${catalina.base}/webapps/myapp/WEB-INF/lib/jna.jar" {
   permission java.security.AllPermission;
};

grant codeBase "file:${catalina.base}/webapps/myapp/WEB- INF/lib/lwjplatform.jar" {
   permission java.security.AllPermission;
};

grant codeBase "file:${catalina.base}/webapps/myapp/WEB- INF/lib/lwservlets.jar" {
   permission java.security.AllPermission;
};

grant codeBase "file:/opt/pbis/lib/-" {
   permission java.security.AllPermission;
};

grant codeBase "file:/opt/pbis/lib64/-" {
   permission java.security.AllPermission;
};
```

Protect Web Pages

You can protect access to resources of your web application either programmatically, or declaratively. It is also possible to use a combination of both methods.

Programmatic security provide for the following APIs, which you can use in JSPs and servlets of your web application:

- **isUserInRole()**: This method determines if the currently authenticated user belongs to a specified role. Roles correspond to groups in Active Directory the Linux computer is joined to. For example, if an HTTP request has been successfully authenticated for user valjean who is a member of **MYDOMAIN\domain^users** group, the following expression would return *true*: **request.isUserInRole("MYDOMAIN\domain^users")**. If no user is currently authenticated (for example, if authorization failed or if **isUserInRole** is called from an unrestricted page and the user has not yet accessed a restricted page), **isUserInRole** returns *false*.
- **getRemoteUser()**: This method returns the name of the current user. For example, if the client has successfully logged in as user **valjean**, **request.getRemoteUser()** would return *valjean*. If no user is currently authenticated (for example, if authorization failed or if **isUserInRole** is called from an unrestricted page and the user has not yet accessed a restricted page), **getRemoteUser** returns *null*.
- **getUserPrincipal()**: This method returns the current username wrapped inside a java.security.Principal object. If no user is currently authenticated, **getUserPrincipal** returns *null*. The Principal object contains little information beyond the username (available with the **getName** method). Likewise extends the standard Principal class to provide access to additional principal information through such methods as **getUserId()**, **getPrimaryGroupId()**, **getHomeDirectory()**, **getSecurityIdentifier()**, **getGecos()**, **getShell()**, **getDistinguishedName()**, **getPrincipalName()**, **isLocalUser()**. This information can be accessed after casting the Principal instance to LikewiseUser class:

```
Principal p = request.getUserPrincipal();
if(p != null) {
  LikewiseUser lwUser = (LikewiseUser) p;
}
```

Declarative security model enables you to describe access control logic directly in the **web.xml** file. For instance, the following **web.xml** sample illustrates how to protect all the web pages in your application so they are accessible only to users who belong to the **MYDOMAIN\domain^users** group.

```
<security-role>
<role-name>MYDOMAIN\domain^users</role-name>
</security-role>
<security-constraint>
  <display-name>Likewise Security Constraint</display-name>
  <web-resource-collection>
     <web-resource-name>Protected Area</web-resource-name>
     <url-pattern>/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
     <role-name>MYDOMAIN\domain^users</role-name>
  </auth-constraint>
</security-constraint>
```

Configure the JAAS Module

AD Bridge Enterprise uses Kerberos5 JAAS module for user authentication. This section explains how to set up the JAAS module to complete the integration of your application server.

1. Create a file named **/opt/p<is/share/config/jaas.policy** and add the following lines to it:

```
grant Principal * * {
permission java.security.AllPermission "/*";
};
```

2. Create a file name **/opt/p<is/share/config/login.conf** and add the following lines to it:

```
Jaas {
com.likewise.auth.jaas.LikewiseLoginModule sufficient;
};
```

3. Add the following java parameters to the command line of the script that starts your application server:

```
-Djava.security.auth.login.config=/opt/p<is/share/config/login.conf
-Djava.security.auth.policy=/opt/p<is/share/config/jaas.policy
```

Restart the Tomcat Server

Restart your application server. For example:

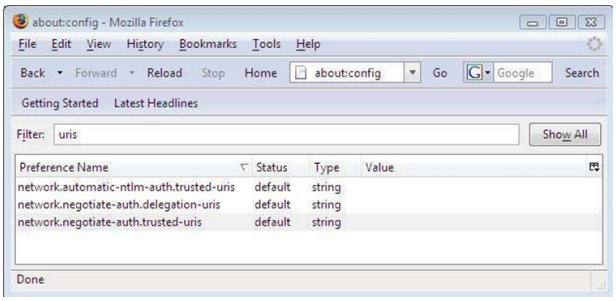
```
/etc/init.d/tomcat restart
```

Set up Firefox and Internet Explorer

Configure Firefox for SSO

To set up Firefox for single sign-on, you must turn on the Simple and Protected GSS-API Negotiation Mechanism, or SPNEGO, to negotiate authentication with Kerberos.

1. Launch Firefox.
2. In the **Go** box, type **about:config**, and then click **Go**.
3. In the **Filter** box, type **uris**.



4. Double-click **network.negotiate-auth.trusted-uris**, enter a comma-separated list of URL prefixes or domains that are permitted to engage in SPNEGO authentication with the browser, and then click **OK.**

5. Double-click **network.negotiate-auth.delegation-uris**, enter a comma-separated list of the sites for which the browser may delegate user authorization to the server, and then click **OK**.

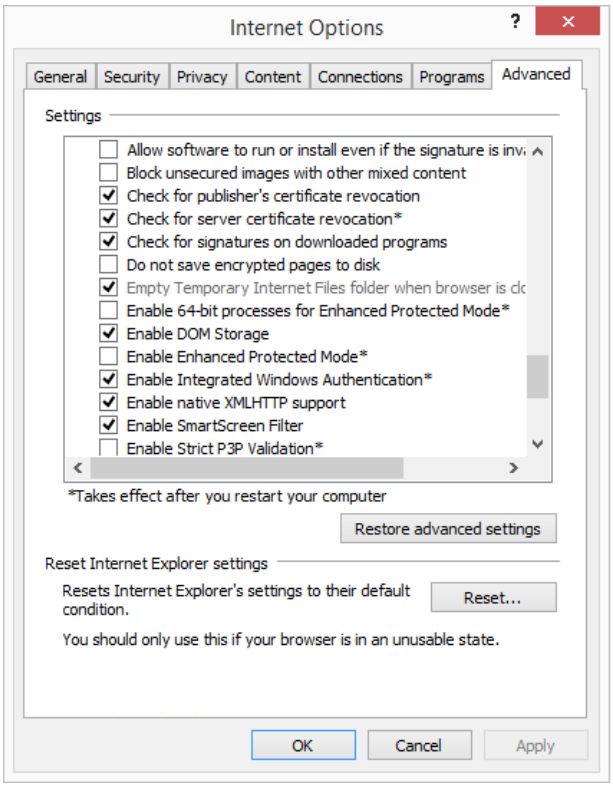
For more information on how to configure Firefox for SSO, please see Enable NTLM Single Sign On in Firefox at https://code.adonline.id.au/enable-ntlm-single-sign-on-in-firefox.

Configure Internet Explorer

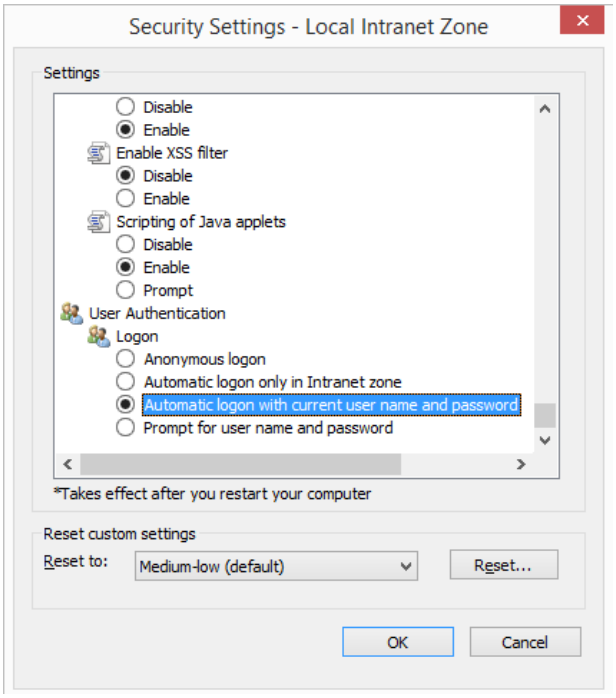
Here's how to configure Internet Explorer 7.0 to use SPNEGO and Kerberos. The settings for other versions of IE might vary; see your browser's documentation for more information.

1. Start Internet Explorer 7.0.
2. On the **Tools** menu, click **Internet Options**.

3. Click the **Advanced** tab and make sure that the **Enable Integrated Windows Authentication** box is checked.



4. Click the **Security** tab.
5. Select a zone. For example, **Local intranet**, and then click **Custom level**.
6. In the **Settings** list, under **User Authentication**, select one of the following:
   - **Automatic logon with current user name and password for a trusted site**
   - **Automatic logon only in Intranet zone for a site you added to IE's list of Intranet sites**



For more information, see your browser's documentation.

7. Return to the **Security** tab for **Internet Options** and set your web server as a trusted site.
8. Restart Internet Explorer.

Test Authentication

The first test is to determine if Integrated Windows Authentication is working for all domain users. As depicted in the above configuration examples, protect your web pages to be accessible by members of **MYDOMAIN\domain^users**.

> ***Note:*** *Use Internet Explorer because it supports Integrated Windows Authentication. Verify that the* ***Enable Integrated Windows Authentication*** *box is checked in the* ***Internet Explorer Internet Options*** *dialog on the* ***Advanced*** *tab. Also, make sure the target server is allowed to be trusted in the* ***Intranet Zone****.*

To Single-Sign-On for a domain user:

1. Log on to a Windows computer that is joined to the same domain you joined your Linux or Unix system to. Log on as a domain user.
2. Access the protected web site from Internet Explorer using the host name. This should use Kerberos authentication if the DNS settings for the client and server are configured accurately.
3. Access the protected web site using the IP address of the server because doing so will result in the Internet Explorer using NTLM for the server. The authentication should succeed without a need to provide a user name and password.

Troubleshoot SSO with Active Directory in a Java Web Application

In the case of authentication failure, there are some diagnostic tools that may help diagnose a problem.

Apache Tomcat Log File

The Apache Tomcat log file may contain helpful information as to the nature of the problem. Typically the Tomcat logs are located under **${CATALINA_HOME}/logs**. It is possible to set the **java.security.debug** variable in the Tomcat environment to elevate the log level and to help check for security issues.

Authentication Groups

Use **/opt/likewise/bin/lw-list-groups-for-user** to list the groups the current user belongs to.

The Microsoft KLIST utility

The Microsoft KLIST utility will help determine whether Internet Explorer obtained a Kerberos ticket for your web server.

Integrated Windows Authentication requires that the client obtains a Kerberos ticket from Active Directory.

To check whether the web browser obtained a Kerberos ticket for your web server:

1. Log on to a Windows computer that is joined to the domain. Log on as a domain user.
2. Access the protected web site from Internet Explorer using the server name in the URL rather than the IP address. For example: https://myserver.
3. From a command prompt run the **KLIST** command. This will show a dialog displaying all Kerberos tickets.
4. Look for the name of your domain in the list of the tickets. Under your domain look for the service principal. It will like this example: **HTTP/myserver.mydomain.com**.

KLIST Linux or Unix Utility

The **klist** Linux or Unix utility, part of the **krb5-client** package, may be used to check the Kerberos keytab file on the Linux or Unix system. The output of this command will display all service principal tickets that are contained in the keytab file. This command may be unavailable on some systems. To use **klist** to examine the contents of a Kerberos keytab file:

> ***Note:*** *Replace* ***myserver*** *with your own server name and* ***mydomain.com*** *with your domain name.*

1. Locate the Kerberos keytab file for the protected web site.

> ***Note:*** *You may need to find the* ***Krb5Keytab*** *directive in the Apache configuration file.*

2. Execute the **klist –k <keytab file name>** command: **klist -k krb5_myserver.keytab**.
3. Verify that the correct service principal names are displayed. The names to look for are **HTTP/myserver@MYDOMAIN.COM** and **HTTP/myserver.mydomain.com@MYDOMAIN.COM**. It is normal to see multiple entries for the same name.

Example output:

```
# klist -k krb5_myserver.keytab
Keytab name: FILE:krb5_myserver.keytab KVNO Principal
---- --------------------------------------------------------
6 HTTP/myserver@MYDOMAIN.COM
6 HTTP/myserver@MYDOMAIN.COM
6 HTTP/myserver@MYDOMAIN.COM
6 HTTP/myserver.mydomain.com@MYDOMAIN.COM
6 HTTP/myserver.mydomain.com@MYDOMAIN.COM
6 HTTP/myserver.mydomain.com@MYDOMAIN.COM
```

If the service principal names are not correct, go back to the Generate Kerberos keytab file step above to generate a new Kerberos keytab file.

Common Issues

As authentication problems may be difficult to diagnose, start with double checking all the configuration parameters including the validity of the generated Kerberos keytab file.

If all the configuration parameters appear to be correct, then examine the list of common problems below.

Problem	Explanation
System clock out of sync	For Kerberos authentication to work, the system clocks on all involved systems must not be more than 5 minutes apart. Make sure that the time on the Active Directory server, Linux or Unix web server and the client are synchronized.
User accessing the web site is not in the configured group	Check the user's group membership using **/opt/likewise/bin/lw-list- groups-for-user**.

Problem	Explanation
Internet Explorer web browser does not consider the URL as part of the Local Intranet zone	This problem is common if the web site is accessed using a URL that includes the full domain name such as https://myserver.mydomain.com. Internet Explorer will only try to obtain Kerberos tickets for websites that are in the Local Intranet zone. Try accessing the web site using just the server name, for example https://myserver. Alternatively, you can add the URL to a list of Local Intranet sites using the **Sites**/**Advanced** buttons in the **Internet Options** dialog on the **Security** tab.
Service Principal Name of the web site is mapped to more than one object in the Active Directory	This is a rather rare problem, but very difficult to diagnose because of lack of a good error that is returned to the web server. This can happen if the **ktpass** Windows utility was used on the Domain Controller to generate a Kerberos keytab file. To diagnose this problem, log on to the Active Directory Domain Controller and open the Event Viewer. Look for event of **type=Error**, **source=KDC**, and **event ID=11**. The text of the event will be similar to this message: *There are multiple accounts with name HTTP/myserver.mydomain.com of type DS_SERVICE_PRINCIPAL_NAME.* Resolving this problem will require locating the computer or user objects used to map the service principal name in the Active Directory. The **Active Directory Users and Computers** MMC snap-in allows running custom LDAP queries. Use an LDAP query similar to **(servicePrincipalName=HTTP/myserver.mydomain.com)** to locate the Active Directory objects. Once objects are located then the spurious User object may be deleted. If the object cannot be deleted then use the ADSI Edit MMC snap-in to manually remove the **HTTP/myserver.mydomain.com** string from the **servicePrincipalName** object property.

Configure AD Bridge and Apache for SSO

This topic describes how to configure AD Bridge Enterprise and the Apache HTTP Server to provide single sign-on authentication through Active Directory with Kerberos 5. The instructions assume that you know how to administer Active Directory, the Apache HTTP Server, and computers running Linux.

Single sign-on for the Apache HTTP server uses the Simple and Protected GSS-API Negotiation Mechanism, or SPNEGO, to negotiate authentication with Kerberos. SPNEGO is an Internet standard documented in RFC 2478 and is commonly referred to as the *negotiate authentication protocol*. The AD Bridge Enterprise **mod_auth_kerb** module lets an Apache web server running on a Linux or Unix system authenticate and authorize users based on their Active Directory domain credentials.

For more information, please see the following articles:

- "Configure a Microsoft Browser for SSO" on page 28.
- "Troubleshoot Single Sign-on and Kerberos Authentication" on page 30.

Prerequisites

- AD Bridge Enterprise installed on the Linux computer running your Apache HTTP Server
- The Apache module ships with the AD Bridge Enterprise agent and is located in either **/opt/pbis/lib64/** or **/opt/pbis/lib/**
- The Linux or Unix computer that is hosting the Apache web server is joined to Active Directory
- An Apache HTTP Server 2.0, 2.2, or 2.4 that supports dynamically loaded modules

To check whether your Apache web server supports dynamically loaded modules, execute the following command and verify that **mod_so.c** appears in the list of compiled modules: **/usr/sbin/httpd -l** or **/usr/sbin/apache2 -1**.

```
Compiled in modules:
  core.c
  prefork.c
  http_core.c
  mod_so.c
```

For Apache installations that are compiled from the source code, make sure that **--enable-module=so** is specified when **./configure** is executed: **./configure --enable-module=so**.

Configure Apache HTTP Server for SSO on RHEL

The following instructions show how to configure AD Bridge Enterprise and Apache for SSO on a Red Hat Enterprise Linux computer. The steps vary by operating system and by Apache version. Ubuntu, in particular, uses **apache2** or **httpd** for commands, the name of the daemon, the configuration directory, the name of the configuration file, etc.

IMPORTANT!

Configuring web servers is complex. Implement and test your configuration in a test environment first. Before you change your web server's configuration:

- *Read the Apache HTTP Server documentation at http://httpd.apache.org/docs/*
- *Read the mod_auth_kerb documentation at http://modauthkerb.sourceforge.net/configure.html.*
- *Back up a file before applying any changes.*

1. Determine whether your Apache server is 2.0, 2.2 or 2.4 by running one of the following commands:
   - **/usr/sbin httpd -v**
   - **/usr/sbin/apache2 -1**

   Example output:

```
Server version: Apache/2.4.6 (Red Hat Enterprise Linux)
Server built:   Aug  3 2016 08:33:27
```

2. Edit the Apache configuration file to add a directive to load the AD Bridge Enterprise **auth_kerb_module** for your version of Apache. Use one of the following:

```
/etc/httpd/conf/httpd.conf
```

```
/etc/apache2/apache2.conf
```

   Since this Red Hat computer is running Apache 2.4.6, the 2.4 version of the module is added, as demonstrated in the following example output:

```
LoadModule auth_kerb_module /opt/pbis/lib64/apache2.4/mod_auth_kerb.so
```

3. In the configuration file, configure authentication for a directory. Example:

```
<Directory "/var/www/html/secure">
 Options Indexes MultiViews FollowSymLinks
 Order deny,allow
 Deny from all
 Allow from 192.0.0.0/8
 AuthType Kerberos
 AuthName "Kerberos Login"
 KrbAuthRealms EXAMPLE.COM
 krb5Keytab /etc/httpd/httpd.keytab
 AllowOverride None
 Require valid-user
</Directory>
```

4. Restart the web server, using the appropriate command for your Apache version:

```
systemctl restart httpd.service
```

```
systemctl restart apache2.service
```

Tip: *You can require that a user be a member of a security group to access the Apache web server by replacing* ***Require valid-user*** *with* ***Require unix-group name-of-your-group****, as shown in the example below. To control group access by requiring group membership, however, you must first install and load* ***mod_authz_unixgroup****. For instructions on how to set up* ***mod_authz_unixgroup****, see https://code.google.com/archive/p/mod-auth-external/wikis/ModAuthzUnixGroup.wiki.*

```
<Directory "/var/www/html/secure">
Options Indexes MultiViews FollowSymLinks
Order deny,allow
Deny from all
Allow from 192.0.0.0/8
AuthType Kerberos
AuthName "Kerberos Login"
KrbAuthRealms EXAMPLE.COM
Krb5Keytab /etc/httpd/httpd.keytab
AllowOverride None
Require unix-group example\linuxusers
</Directory>
```

Configure your web server for Secure Socket Layer (SSL).

For instructions on configuring your web server for SSL, please see the Apache HTTP Server documentation at http://httpd.apache.org/docs/current/.

IMPORTANT!

If SSO fails and you have not turned on SSL, your server will prompt you for an ID and password, which will be sent in clear text. SSL encrypts all data that passes between the client browser and the web server. SSL can also perform Basic Authentication securely, providing a fallback mechanism if Kerberos authentication fails. Using SSL is especially important if the protected website also needs to be accessible from outside the corporate network. For more information, please see http://modauthkerb.sourceforge.net/configure.html.

In Active Directory, create a user account for the Apache web server in the same OU (or Cell, with AD Bridge Enterprise) to which the Linux computer hosting the web server is joined. Set the password of the user account to never expire. In the examples that follow, the user account for the Apache web server is named **httpUser**.

On the domain controller, create an RC4-HMAC keytab for the Apache web server using Microsoft's **ktpass** utility. The keytab that you must create can vary by Windows version.

For information on ktpass, please see Ktpass Syntax at https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc776746(v=ws.10).

Example:

```
C:\>ktpass /out keytabfile /princ HTTP/rhel7.example.com@EXAMPLE.COM /pass password /mapuser
example\httpUser /ptype KRB5_NT_PRINCIPAL
Targeting domain controller: dc1.example.com
Using legacy password setting method
Successfully mapped HTTP/rhel7.example.com to httpUser.
Key created.
Output keytab to keytabfile:
Keytab version: 0x502
keysize 80 HTTP/rhel7.example.com@EXAMPLE.COM ptype 0 (KRB5_NT_UNKNOWN) vno 3 etype 0x17 (RC4-
HMAC) keylength 16 (0x2998807dc299940e2c6c81a08315c596)
```

1. Use secure FTP or another method to transfer the keytab file to the Linux computer that hosts your Apache web server and copy the file to the location specified in your **<Directory>** configuration in **httpd.conf**. For example, using the configuration shown in Step 3 above, copy the keytab file to **/etc/apache2/http.ktb**.
2. Set the permissions of the keytab file to be readable by the ID under which the Apache web server runs and no one else.

IMPORTANT!

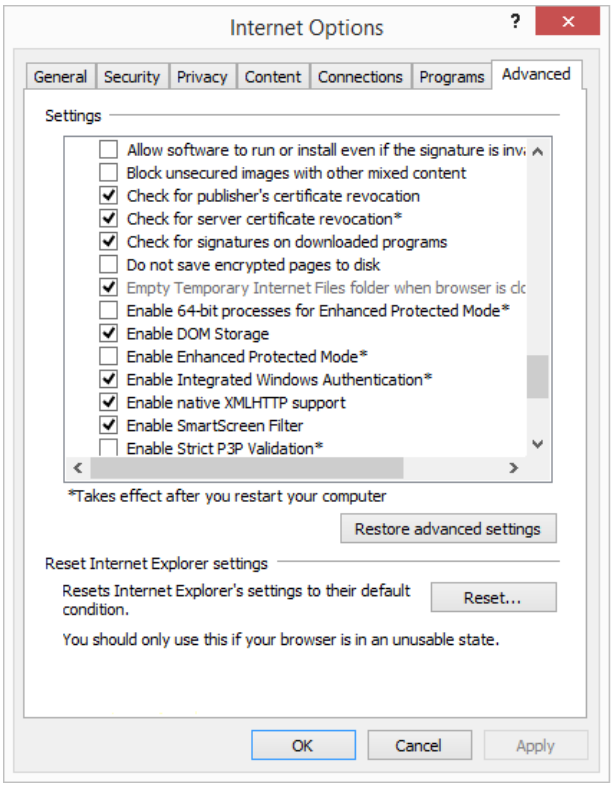
The Kerberos keytab file is necessary to authenticate incoming requests. It contains an encrypted, local copy of the host's key and, if compromised, might allow unrestricted access to the host computer. It is therefore crucial to protect it with file-access permissions.

Configure a Microsoft Browser for SSO

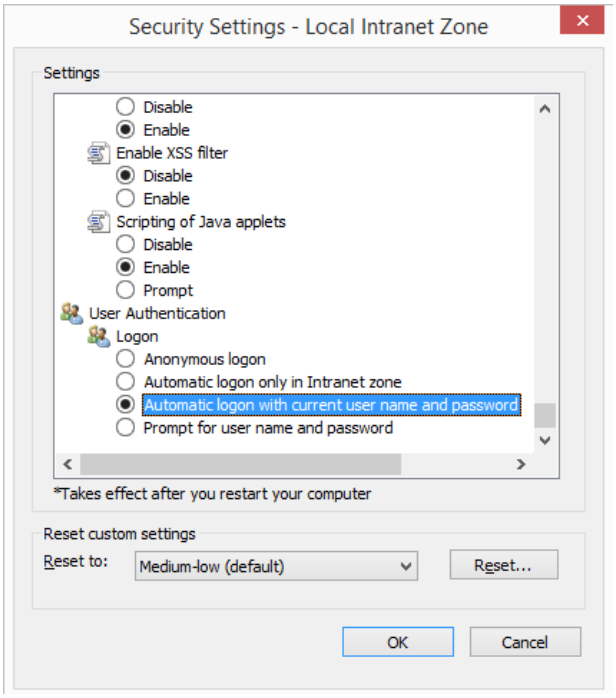
You can configure Microsoft Internet Explorer or Microsoft Edge to use SPNEGO and Kerberos. The settings might vary depending on the browser. See the browser's documentation for more information.

1. Start **Internet Explorer**.
2. On the **Tools** menu, click **Internet Options**.

3. Click the **Advanced** tab and check the **Enable Integrated Windows Authentication** box.



4. Click the **Security** tab.
5. Select a zone, for example, **Local intranet**, and then click **Custom level**.
6. In the **Settings** list, under **User Authentication**, click **Automatic logon with current user name and password** for a trusted site, or **Automatic logon only in Intranet zone** for a site you added to IE's list of Intranet sites. For more information, see the browser's documentation.



7. Return to the **Security** tab for **Internet Options** and set your web server as a trusted site.
8. Restart the browser.

Troubleshoot Single Sign-on and Kerberos Authentication

The following tools and procedures can help diagnose and resolve problems with Kerberos authentication when using the Apache HTTP Server for single sign-on (SSO).

Apache Log File

The location of the Apache error logs is specified in the Apache configuration file under the **ErrorLog** directive. The default value is, depending on your Apache version, one of the following:

- **/var/log/httpd/**
- **/var/log/apache2/**

Klist Utility

You can use the **klist** utility in **/opt/pbis/bin/klist** to check the Kerberos keytab file on a Linux or Unix computer. The command shows all the service principal tickets contained in the keytab file so you can verify that the correct service principal names appear.

Confirm that **HTTP/rhel7@EXAMPLE.COM** and **HTTP/rhel7.example.com@EXAMPLE.COM** appear in the list. It is normal to see multiple entries for the same name.

Example:

```
$ /opt/pbis/bin/klist -k /etc/httpd/httpd.ktb
Keytab name: WRFILE:/etc/httpd/httpd.ktb
KVNO Principal
---- --------------------------------------------------------------------------
5 HTTP/rhel7.example.com@EXAMPLE.COM
```

If your service principal names are incorrect, generate a new Kerberos keytab file.

Tip: *Because you cannot store credentials for more than one principal in a Kerberos credentials cache at a time, you must maintain two or more credential caches by using the* ***KRB5CCNAME*** *environment variable and then switch to the cache that you want to use. To use an alternate Kerberos cache with AD Bridge Enterprise, for example, you could execute the following sequence of commands as root:*

```
[root@oracle1 ~]# KRB5CCNAME=/var/lib/pbis/krb5cc_lsass
[root@oracle1 ~]# export KRB5CCNAME
[root@oracle1 ~]# klist
Ticket cache: FILE:/var/lib/pbis/krb5cc_lsass
```

Confirm User Received an HTTP Ticket

Klist can be used on the current user to verify that they receive a service ticket for HTTP.

Run **Klist** on Linux and UNIX systems running AD Bridge or on Windows from the command prompt.

Linux klist:

```
apacheuser@rhel7:/home/apacheuser$ /opt/pbis/bin/klist
Ticket cache: FILE:/tmp/krb5cc_2066220575
Default principal: apacheuser@EXAMPLE.COM
   Valid starting     Expires            Service principal
   04/05/17 11:46:28  04/05/17 21:46:28
      krbtgt/EXAMPLE.COM@EXAMPLE.COM
      renew until 04/05/17 23:46:28
   04/05/17 11:46:28  04/05/17 21:46:28
      host/rhel7.example.com@EXAMPLE.COM
      renew until 04/05/17 23:46:28
   04/05/17 11:46:28  04/05/17 21:46:28  ldap/dc1.example.com@EXAMPLE.COM
      renew until 04/05/17 23:46:28
   04/05/17 11:46:28  04/05/17 21:46:28  HTTP/rhel7.example.com@EXAMPLE.COM
      renew until 04/05/17 23:46:28
```

Windows klist:

```
C:\>klist
Current LogonId is 0:0x816aded2
Cached Tickets: (5)
#0> Client: apacheuser @ EXAMPLE.COM
   Server: krbtgt/EXAMPLE.COM @ EXAMPLE.COM
   KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
   Ticket Flags 0x60a00000 -> forwardable forwarded renewable pre_authent
   Start Time: 4/5/2017 9:09:52 (local)
   End Time:   4/5/2017 19:09:52 (local)
   Renew Time: 4/12/2017 9:09:52 (local)
   Session Key Type: AES-256-CTS-HMAC-SHA1-96
   Cache Flags: 0x2 -> DELEGATION
   Kdc Called: dc1.example.com
#1> Client: apacheuser @ EXAMPLE.COM
   Server: krbtgt/EXAMPLE.COM @ EXAMPLE.COM
   KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
   Ticket Flags 0x40e00000 -> forwardable renewable initial pre_authent
   Start Time: 4/5/2017 9:09:52 (local)
   End Time:   4/5/2017 19:09:52 (local)
   Renew Time: 4/12/2017 9:09:52 (local)
   Session Key Type: AES-256-CTS-HMAC-SHA1-96
   Cache Flags: 0x1 -> PRIMARY
   Kdc Called: dc1.example.com
#2> Client: apacheuser @ EXAMPLE.COM
   Server: HTTP/rhel7.example.com @ EXAMPLE.COM
   KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
   Ticket Flags 0x40a00000 -> forwardable renewable pre_authent
   Start Time: 4/5/2017 9:15:36 (local)
   End Time:   4/5/2017 19:09:52 (local)
   Renew Time: 4/12/2017 9:09:52 (local)
   Session Key Type: RSADSI RC4-HMAC(NT)
   Cache Flags: 0
   Kdc Called: dc1.example.com
#3> Client: apacheuser @ EXAMPLE.COM
   Server: ldap/DC13.example.com/example.com @ EXAMPLE.COM
```

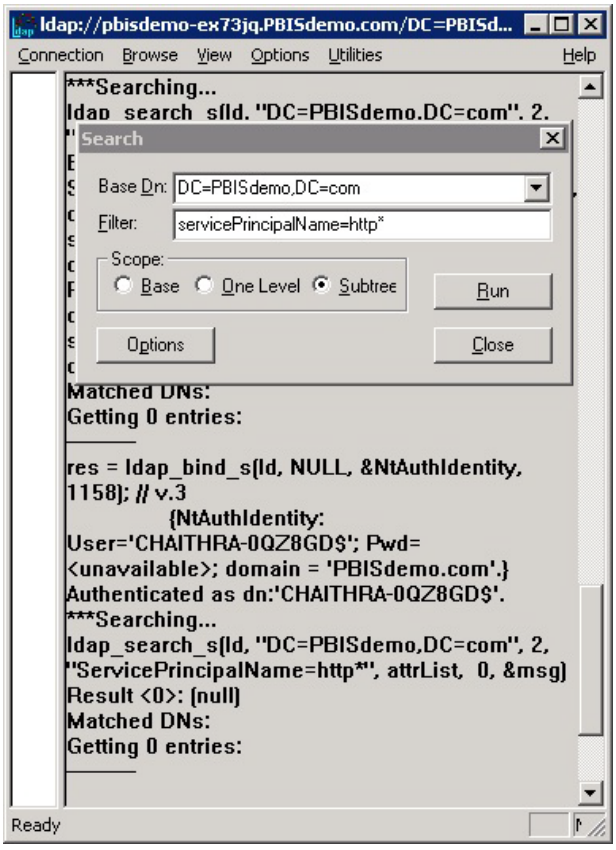
```
    KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
    Ticket Flags 0x40a40000 -> forwardable renewable pre_authent ok_as_delegate
    Start Time: 4/5/2017 9:09:53 (local)
    End Time:   4/5/2017 19:09:52 (local)
    Renew Time: 4/12/2017 9:09:52 (local)
    Session Key Type: AES-256-CTS-HMAC-SHA1-96
    Cache Flags: 0
    Kdc Called: dc1.example.com
#4> Client: apacheuser @ EXAMPLE.COM
    Server: cifs/DC11 @ EXAMPLE.COM
    KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
    Ticket Flags 0x40a40000 -> forwardable renewable pre_authent ok_as_delegate
    Start Time: 4/5/2017 9:09:52 (local)
    End Time:   4/5/2017 19:09:52 (local)
    Renew Time: 4/12/2017 9:09:52 (local)
    Session Key Type: AES-256-CTS-HMAC-SHA1-96
    Cache Flags: 0
    Kdc Called: dc1.example.com
```

Resolve Common Problems

Authentication problems can be difficult to diagnose. First, check all the configuration parameters, including the validity of the keytab file. Second, review the common problems in the following table.

Problem	Solution
The system's clock is out of sync.	The Kerberos standard requires that system clocks be no more than 5 minutes apart. Make sure that the system clocks on the Active Directory domain controller, the Linux or Unix web server, and the client are synchronized.
The user accessing the website is not on the **require** list.	If the Kerberos ticket was obtained on the client or the user correctly entered his credentials during the Basic Authentication prompt, it might be because authentication worked but the authorization failed. If so, the Apache **error_log** will contain a line like this: *access to / failed, reason: user EXAMPLE\\user not allowed access* Add the user to the **require user** directive or add the user's group to the **require group** directive.
The user accessing the website is logged on the wrong domain.	If the client user is logged on a domain different from the domain of the web server, one of two things will happen: 1. If the **KrbMethodK5Passwd** directive is set to **on**, or was not specified and thus defaults to **on**, the user will be prompted for credentials. 2. If **KrbMethodK5Passwd** is set to **off**, authentication will fail and the **Authorization Required** page will be displayed.

Problem	Solution
Internet Explorer does not consider the URL to be part of the Local Intranet zone or the Trusted sites.	This problem commonly occurs when the website is accessed by using a URL that includes the full domain name, such as https://myserver.example.com. Internet Explorer tries to obtain Kerberos tickets only for websites that are in the Local Intranet zone. Try to access the website by using only the server name, for example https://myserver. Or, you can add the URL to a list of Local Intranet sites or the trusted sites by changing your options in Internet Explorer.
The service principal name of the website is mapped to more than one object in the Active Directory.	Although this problem is rare, it is difficult to diagnose because the error messages are vague. The problem can occur after the **ktpass** utility was used repeatedly to generate a Kerberos keytab file for the web server. To check for this problem, log on your Active Directory domain controller and open the Event Viewer. Look for an event of **type=Error**, **source=KDC**, and **event ID=11**. The text of the event will be similar to the message below: *There are multiple accounts with name HTTP/myserver.example.com of type DS_SERVICE_PRINCIPAL_NAME.* To fix the problem, find the computer or user objects that were used to map the service principal name in Active Directory and then use the ADSI Edit to manually remove the **HTTP/myserver.example.com** string from the **servicePrincipalName** object property. Below the table is a screen shot that provides an example of how to find an object named **HTTP** by using LDAP.



Resolve Kerberos Library Mismatch

Because some operating systems, such as the 64-bit version of Red Hat Enterprise Linux 5, use an outdated version of **/lib/libcom_err.so**, the AD Bridge Enterprise authentication agent cannot the locate the proper system library, leading to an error that looks like this:

```
httpd: Syntax error on line 202 of /etc/httpd/conf/httpd.conf:Cannot load /opt/pbis/apache/2.2/mod_
auth_kerb.so into server:
/opt/pbis/lib/libcom_err.so.3: symbol krb5int_strlcpy, version
krb5support_0_MIT not defined in file libkrb5support.so.0 with link time reference
```

Solution: Force the **httpd** daemon to use the AD Bridge Enterprise krb5 libraries by opening the startup script for the Apache HTTP Server: **/etc/init.d/httpd** Additionally, add the path to the AD Bridge Enterprise Kerberos libraries on the line that starts Apache. The line that starts the daemon can vary by operating system. Example on a 64-bit system: **LD_LIBRARY_PATH=/opt/pbis/lib64 LANG=$HTTPD_LANG daemon $httpd $OPTIONS**

> ***Note:*** *This modification changes the version of the Kerberos libraries that are used by the Apache HTTP Server. The change might result in compatibility issues with other modules of Apache that use Kerberos.*

Configure SSH for AD Bridge Integration

Enable PAM for SSH

For your Active Directory account to work with SSH:

- Enable **UsePAM** in **sshd_config**
- Link **sshd** application to the PAM libraries

1. Determine which **sshd** is running by executing the following command:

```
bash-3.2# ps -ef | grep sshd
    root  8199     1  0  Feb  6  ?         0:00 /opt/ssh/sbin/sshd
    root  2987  8199  0  Mar  3  ?         0:04 sshd: root@notty
    root 24864  8199  0 12:16:25 ?         0:00 sshd: root@pts/0
    root  2998  8199  0  Mar  3  ?         0:05 sshd: root@notty
    root 24882 24880  0 12:16:54 pts/0     0:00 grep sshd
```

2. Either use **lsof** to find out which configuration file it is reading or start it up with debugging to figure out the default path. Example:

```
username@computer:~$ /usr/sbin/sshd -dd -t
   debug2: load_server_config: filename /etc/ssh/sshd_config
   debug2: load_server_config: done config len = 664
   debug2: parse_server_config: config /etc/ssh/sshd_config len 664
   debug1: sshd version OpenSSH_5.1p1 Debian-3ubuntu1
   Could not load host key: /etc/ssh/ssh_host_rsa_key
   Could not load host key: /etc/ssh/ssh_host_dsa_key
```

3. Verify that **UsePAM** is enabled in the config file. As a best practice, make a backup copy of the configuration file before you change it.
4. Run **ldd** on **sshd** to make sure it links with **libpam**. Here is an example from an IA64 HP system:

```
bash-3.2# ldd /opt/ssh/sbin/sshd
        libpam.so.1 =>  /usr/lib/hpux64/libpam.so.1
        libdl.so.1 =>   /usr/lib/hpux64/libdl.so.1
        libnsl.so.1 =>  /usr/lib/hpux64/libnsl.so.1
        libxnet.so.1 => /usr/lib/hpux64/libxnet.so.1
        libsec.so.1 =>  /usr/lib/hpux64/libsec.so.1
        libgssapi_krb5.so =>    /usr/lib/hpux64/libgssapi_krb5.so
        libkrb5.so =>   /usr/lib/hpux64/libkrb5.so
        libpthread.so.1 =>      /usr/lib/hpux64/libpthread.so.1
        libc.so.1 =>    /usr/lib/hpux64/libc.so.1
        libxti.so.1 =>  /usr/lib/hpux64/libxti.so.1
        libxti.so.1 =>  /usr/lib/hpux64/libxti.so.1
        libm.so.1 =>    /usr/lib/hpux64/libm.so.1
```

```
        libk5crypto.so =>       /usr/lib/hpux64/libk5crypto.so
        libcom_err.so =>        /usr/lib/hpux64/libcom_err.so
        libk5crypto.so =>       /usr/lib/hpux64/libk5crypto.so
        libcom_err.so =>        /usr/lib/hpux64/libcom_err.so
        libdl.so.1 =>   /usr/lib/hpux64/libdl.so.1
```

Configure GSSAPI for SSH

Logging onto a system with keys does not provide that system with the means of getting a PAC from the domain controller. Without a PAC there is no group membership information for the user. Automated Kerberos ticket renewal will also be unavailable. So, when the ssh login hits the login restrictions in the account phase as it tests for the group memberships, it will not find the user's group information, causing an ssh error like this:

```
Not in an Allowed Group!
```

A workaround is to have each user log in once with a password. Subsequent logins with keys should work until the AD cache is flushed, after which the user will have to log in again.

Check the Configuration of SSH for SSO

Although AD Bridge Enterprise automatically configures OpenSSH to support SSO through Kerberos using GSSAPI, it is worthwhile to review how AD Bridge Enterprise does. Since you might need to configure or troubleshoot other applications for SSO, understanding the process will make it easier to apply the technique to other applications.

Note: *Not all versions of OpenSSH support Kerberos. Versions older than 4.2p1 might not work or might work improperly.*

SSH Service Principal Name

The first thing that needs to be considered is the Kerberos service principal name (SPN) used by **ssh** and **sshd**. The SPN is a string that identifies the service for which an authentication ticket is to be generated. In the case of **ssh**, the SPN has the form:

```
host/<server name>@<REALMNAME>
```

For example, when a user uses **ssh** to connect to a computer named **fozzie.mycorp.com**, the **ssh** program requests a service ticket for the SPN:

```
host/fozzie.example.com@EXAMPLE.COM
```

The Kerberos realm is the computer's domain name in uppercase letters.

System Keytab Generation

In order for Microsoft Active Directory to generate a Kerberos ticket for this SPN, a service account must exist for it. Additionally, a keytab must be created for the service account and placed on the sshd server. AD Bridge Enterprise completely automates this operation. When a Linux or Unix computer is joined to AD, a machine account is created for the computer. If the computer is called **fozzie**, a machine account called **fozzie$** is created in AD. AD Bridge Enterprise then automatically creates a keytab for the SPN and places it in the standard system location (typically, **/etc/krb5.keytab**).

User Keytab Generation

When the user runs the **ssh** program and OpenSSH determines that it will use Kerberos authentication, it will need to access a keytab for the user so that it can obtain a service ticket for the service or computer to which it is trying to connect. This keytab must be created using the user's account name and password. Manually, this can be performed by using the **kinit** utility. AD Bridge Enterprise, however, does it automatically when the user logs on the computer. On most systems, the user keytab is placed in the **/tmp** directory and named **krb5cc_UID** where **UID** is the numeric user ID assigned by the system.

Configure OpenSSH

AD Bridge Enterprise automatically configures OpenSSH at both the client and server computer. On the client, the **ssh_config** file (typically in **/etc/ssh/ssh_config**) is modified. On the server, **ssh_config** (typically in **/etc/ssh/ssh_config**) is modified. AD Bridge Enterprise adds the following lines of code to the right files if they are not already present and if they are required by the system's version of sshd:

In the server, the following lines must be present in **sshd_config**. if you are troubleshooting, make sure these lines are there:

```
GSSAPIAuthentication yes
GSSAPICleanupCredentials yes
```

On the client, the following line must be present in **ssh_config**:

```
GSSAPIAuthentication yes
```

On the client, **GSSAPIDelegateCredentials yes** is an optional setting that instructs the ssh client to delegate the krb5 TGT to the destination machine when SSH single sign-on is used.

In addition, if any of the following options are valid for the system's version of sshd, they are required and configured by AD Bridge Enterprise:

```
ChallengeResponseAuthentication yes
UsePAM yes
PAMAuthenticationViaKBDInt yes
KbdInteractiveAuthentication yes
```

Setting these options to **yes** instructs **ssh** to use the **kbdinteractive** ssh authentication mechanism and allows that mechanism to use PAM, settings that are required for AD Bridge Enterprise to function properly.

For more information, see the man pages for **ssh**, **sshd**, and the comments in the **ssh** and **sshd** configuration files.

Test SSO

With OpenSSH properly configured, demonstrating SSO support is simple: Log on a Linux or Unix machine running AD Bridge Enterprise by using your Active Directory credentials and then use **ssh** to connect to another machine that is also running AD Bridge Enterprise. OpenSSH should establish a connection without prompting for a username or password.

Smart Card Authentication and Troubleshooting

Here is what you need to get started:

- A Linux platform supported by the AD Bridge Enterprise Smart Card service.
- An Active Directory system configured to manage Smart Card logons.
- A Smart Card prepared with Active Directory credentials and a personal identification number to log on to the Linux computer.
- A CCID-compliant Smart Card reader.
- You must install a CCID-compliant Smart Card reader. The readers are available from a variety of manufacturers. Before you buy a reader, check with the vendor to make sure it works with your Linux platform and your type of Smart Card. Follow the setup instructions from the manufacturer of the Smart Card reader.
- AD Bridge Enterprise 8.5.3 or later. When you install AD Bridge Enterprise, you must include the smartcard option.

Note: *AD Bridge Enterprise 8.5.3 or later. ActivIdentity's 32-bit driver ActiveClient is no longer installed.*

- Linux computers (64-bit) need a 3rd party Smart Card driver installed. For example, OpenSC provides support for PIV II Smart Cards.

Supported Linux Platforms

The AD Bridge Enterprise Smart Card service supports 64-bit versions of Red Hat Enterprise Linux 6.x or later.

To check the version of your Red Hat computer: **cat /etc/redhat-release**.

Example output:

```
[auser@rhel7 ~]# cat /etc/redhat-release
Red Hat Enterprise Linux Client release 7.3 (Maipo)
```

On 64-bit systems, you must install a 3rd party Smart Card driver and Smart Card reader. OpenSC provides **opensc-tool** and **pkcs11-tool** and a PCSC daemon.

Install the Smart Card Service

To install AD Bridge Enterprise to support Smart Cards, you must include the **smartcard** option when you run the installer. If AD Bridge Enterprise is already installed, run the installer again with the **smartcard** option.

Replace **x.x.x.xxxx** with the version and build number indicated in the installer file name: **./pbis-enterprise-x.x.x.xxxx.linux.x86_64.rpm.sh -- --smartcard install**

Verify Smart Card Settings

If OpenSC is used, it is recommended that the following two commands are used to verify the Smart Card reader is installed correctly and certificates on the Smart Card can be read:

- **opensc-tool**: Ensure a Smart Card reader is installed and a token is inserted.
- **pkcs11-tool**: Ensure certificates on the Smart Card are readable.

Verify pcsc-lite is Installed

AD Bridge Enterprise depends on the presence of a package, **pcsc-lite**. To confirm the package is installed, run the following command: **rpm -q pcsc-lite**.

Example

```
root@rhel5d lw]# rpm -q pcsc-lite
pcsc-lite-1.3.1-7
```

When this initial configuration is in place, you are ready to install AD Bridge Enterprise on your Linux computer and add the computer to Active Directory.

For information on installing the AD Bridge Enterprise agent and joining a domain, please see the AD Bridge Enterprise Installation Guide at www.beyondtrust.com/docs/ad-bridge/getting-started/installation.

Alternate pkcs11 Library Location

Using the config tool's **ModuleSearchList** option, set the registry with the directory location of the third party pkcs11 library. **Lwpkcs11d** will reference the registry to determine which library to load. Currently three locations are hardcoded in **lwpkcs11** daemon.

```
/opt/pbis/bin/config --details ModuleSearchList
Name: ModuleSearchList
Description: Determines which pkcs11 module lwpkcs11 daemon uses to access Smart Card functionality.
Type: multistring
Current Values:
  "/usr/lib/libpkcs11.so"
  "/usr/local/lib/libpkcs11.so"
  "/usr/lib64/opensc-pkcs11.so"
Current Value is determined by local policy.
```

Troubleshoot

The following section provides information on troubleshooting the card and reader. Verify:

- Smart Card reader is installed with a Smart Card token
- PKCS11 library is installed
- AD Bridge Enterprise was installed with the **smartcard** option
- The server is joined to a domain
- The Smart Card service is configured to use the installed PKCS11 library
- Smart Card services **lwsc** and **lwpkcs11** are running

Smart Card Diagnostic Tool

A tool is available withAD Bridge that can:

- verify the Smart Card reader installation
- read the contents of the Smart Card

- verify the enrolled Smart Card user certificate Subject Alternate Name (SAN) has the User Principal Name (UPN)

Run the following command to use the tool: **/opt/pbis/bin/sc-test**.

Install OpenSC

OpenSC provides a PCSC driver and several command line tools like **opensc-tool** and **pkcs11-tool**. Restart the server after you install OpensSC: **yum install opensc**.

Plug in the Smart Card Reader

Run the following command. The Smart Card reader should be listed: **lsusb**.

Example output:

```
Bus 002 Device 005: ID 058f:9540 Alcor Micro Corp. AU9540 Smartcard Reader
```

OpenSC Commands

```
# opensc-tool --list-readers
   # Detected readers (pcsc)
   Nr.  Card  Features  Name
   0 Yes Alcor Micro AU9540 00 00
# opensc-tool --reader 0 --name -v
   Connecting to card in reader Alcor Micro AU9540 00 00...
   Using card driver PIV-II  for multiple cards.
   Card name: PIV-II card
```

PKCS 11 commands

List the Contents of the Smart Card

```
pkcs11-tool  --module /usr/lib64/opensc-pkcs11.so  -O
```

It is preferred that the enrolled certificate is in slot ID 1. If not, reference the enrollment workstation and ensure the enrolled certificate is mapped to **PivCert9A**. See above.

Other pkcs11-tool Commands

```
# pkcs11-tool  --module /usr/lib64/opensc-pkcs11.so --list-slots
# pkcs11-tool  --module /usr/lib64/opensc-pkcs11.so  --slot 1 --list-objects
# pkcs11-tool  --module /usr/lib64/opensc-pkcs11.so -l -O --id 4
# pkcs11-tool  --module /usr/lib64/opensc-pkcs11.so  --show-info
# pkcs11-tool  --module /usr/lib64/opensc-pkcs11.so --list-mechanisms
```

Troubleshoot the Smart Card

- Ensure the Smart Card services are running:

```
lwpkcs11
lwsc
```

- Restart the AD Bridge Enterprise server after installing AD Bridge Enterprise with the **--smartcard** option.
- Increase the log level on **lsass** and identify issues in logs.

AD Bridge Password Safe Integration

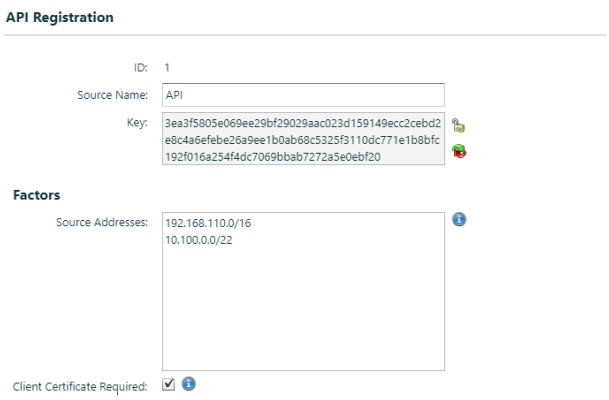
You can add AD Bridge Enterprise agent machines as managed systems to AD Bridge Password Safe.

***Note:** The following instructions assume that Password Safe is successfully installed.*

Configure the API

To configure the API:

1. In the BeyondInsight management console, go to the API Registration configuration page.
2. Add a source name, generate a key, and add IP addresses.
3. Check the **Client Certificate Required** box.



Create an API User

You must create a user account in Password Safe that can use the Password Safe API.

For more information about configuration, refer to the Password Safe Administration Guide at https://www.beyondtrust.com/docs/beyondinsight-password-safe/ps/index.htm.

To create the user account:

1. In the BeyondInsight management console, go to the **Users & Groups** configuration page.
2. Create a group.
3. Set permissions and select at least one Smart Rule.
4. Check the **Enable Application API** box, and then check the box for the application (created in "Configure the API" on page 42).
5. After the group is created, create a user account. Create a user name that will easily identify the user (for example, **apiuser**).

Create a Managed Account

Keep the following in mind when creating the managed account:

- The managed account must be the same as the account in Active Directory.
- The account will be used for the domain join. This user must have permissions to complete a domain join on a Linux/Unix agent.
- On the **Managed Account Settings** page, the user needs the **Enable for API access** box checked.
- The domain information needs to be populated for the domain you want to join.

Configure the Agent

There are two ways to configure the agent.

Regardless of the option chosen, the information in the configuration file needs to be defined. The information is from Password Safe and is defined in the sample template below.

The configuration file is: **djpbpbs.config.template** and is installed in the **/etc/pbis** folder.

Option 1

If the API key created does not require the user to have a certificate, then a user name and password combination can be used in the **djpbp.config.template**. This can be done by defining the API user and password and leaving the **Certificate** section commented out.

Sample djpbps.config.template:

```
[Version]
Template = 1  # Do not modify.
[DomainJoin]
# User account to use for domain join. In addition, this account must
# also be a managed account in Password Safe with API access enabled.
DomainJoinUser = ' '
[PasswordSafe]
# Password Safe URL
ServerUrl = 'https://0.0.0.0.'   # eg. https://server-name
# From Password Safe web console Configuration->Users & Groups->User Group.
# Check box "Enable Application API"
RunAsUser = ''
# Optional. Needed if Password Safe web console shows "User Password Required"
# is checked in Configuration->API Registration. Uncomment if required.
#RunAsUserPassword = ''
# From Password Safe web console Configuration->API Registration->Key
ApiKey = ' ' #eg.
4b2c430dbe2b6aff66b016cc8e11b0f78b4d7cc426d3fd4c53c92a261226fbe8ce8f0b5f42974789210420196c6539135c49
2834123c93ed6f7d53023dfa9a4a'
# Minutes PBPS credential is valid until it expires.
# Optional. Valid range is between 1 and 10079 inclusive.
# Default is 1 minute. Uncomment if required.
#DurationMinutes = 1
# Client Certificate.
# Optional. Needed if Password Safe web console shows "Client Certificate
# Required" is checked in Configuration->API Registration.
# If given, the certificate is stored in /etc/pbis/pbpsClient.pem.
```

```
# Default is not to use the client certificate.
# Uncomment if required.
ClientCertificate = """
-----BEGIN PRIVATE KEY-----
-----END PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----
"""
# CA Certificate
# Optional. Specify if PBIS agent should verify PBPS server.
# If given, the certificate is stored in /etc/pbis/pbpsCA.pem.
# Default is not to verify. Uncomment if required.
#CACertificate = """
#-----BEGIN CERTIFICATE-----
#   ...
#   ...
#   ...
#-----END CERTIFICATE-----
#"""
```

Option 2

Export Password Safe Certificates

Certificates can be exported from the UVM appliance **Maintenance** page.

To export a certificate:

1. Log on to the UVM appliance **Maintenance** page URL (for example, PasswordSafe_URL/Maintenance).
2. Select **Security Settings** from the menu.
3. Select **Export Client Certificate**.

Use Password Safe to Join an Agent to a Domain

After the configuration file is completed with the correct information, the agent machine is ready for integration with Password Safe.

To run the domainjoin with the config file, run it as: **/opt/pbis/bin/domainjoin-cli verbose join --ou test --configFile /tmp/djpbps.config example.com**.

A configuration file only has to be included in the domain leave if the user is using the **–deleteAccount** option. For example, **/opt/pbis/bin/domainjoin-cli leave --configFile /tmp/djpbps.config –deleteAccount**.

If the user is not issuing a domain leave with the **deleteAccount** flag, the configuration file does not need to be specified.

Open a Trouble Ticket With BeyondTrust Technical Support

BeyondTrust provides an online knowledge base, as well as telephone and web-based support.

For BeyondTrust Technical Support contact information, please visit www.beyondtrust.com/support.

Before Contacting BeyondTrust Technical Support

To expedite support, collect the following information to provide to BeyondTrust Technical Support:

- AD Bridge Enterprise version: available in the AD Bridge Enterprise Console by clicking **Help > About** on the menu bar
- AD Bridge Enterprise Agent version and build number
- Linux or Unix version
- Windows or Windows Server version

If you are contacting BeyondTrust Technical Support about one of the following problems, also provide the diagnostic information specified.

Segmentation Faults

Provide the following information when contacting BeyondTrust Technical Support:

- Core dump of the AD Bridge application:

```
ulimit - c unlimited
```

- Exact patch level or exact versions of all installed packages

Program Freezes

Provide the following information when contacting BeyondTrust Technical Support:

- Debug logs
- tcpdump
- An **strace** of the program

Domain-Join Errors

Provide the following information when contacting BeyondTrust Technical Support:

- Debug logs: copy the log file from **/var/log/pbis-join.log**
- tcpdump

All Active Directory Users Are Missing

Provide the following information when contacting BeyondTrust Technical Support:

- Run **/opt/pbis/bin/get-status**
- Contents of **nsswitch.conf**

All Active Directory Users Cannot Log On

Provide the following information when contacting BeyondTrust Technical Support:

- Output of **id <user>**
- Output of **su -c 'su <user>' <user>**
- **lsass** debug logs

> For more information, please see *Generate Debug Logs* in the AD Bridge Troubleshooting Guide at www.beyondtrust.com/docs/ad-bridge/how-to/troubleshoot.

- Contents of **pam.d/pam.conf**
- The sshd and ssh debug logs and syslog

AD Users or Groups are Missing

Provide the following information when contacting BeyondTrust Technical Support:

- The debug logs for lsass
- Output for **getent passwd** or **getent group** for the missing object
- Output for **id <user>** if user
- tcpdump
- Copy of lsass cache file.

Poor Performance When Logging On or Looking Up Users

Provide the following information when contacting BeyondTrust Technical Support:

- Output of **id <user>**
- The lsass debug log
- Copy of lsass cache file.

> For more information about the file name and location of the cache files, please see the AD Bridge Linux Administration Guide at www.beyondtrust.com/docs/ad-bridge/getting-started/linux-admin.

- tcpdump

Generate a Support Pack

The AD Bridge support script will copy system files that AD Bridge needs to function into an archive. This archive can then be sent to support to assist in the investigation.

Installed location:

/opt/pbis/libexec/pbis-support.pl